A Common Configurator Framework for Distributed Design, Collaboration and Verification Across the Full AEC Supply Chain

Dr. Al Fisher

Buro Happold, United Kingdom
al.fisher@burohappold.com

Georgios Tsakiridis
Grimshaw Architects, United Kingdom
georgios.tsakiridis@grimshaw.global

Alessio Lombardi

Buro Happold, United Kingdom
alessio.lombardi@burohappold.com

Dr. Maz Ahmad

The Manufacturing Technology Centre, United Kingdom
maz.ahmad@the-mtc.org

Andy Watts

Grimshaw Architects, United Kingdom
andy.watts@grimshaw.global

Abstract. The wider AEC industry has recently seen a growing number of platform-based approaches, focusing on the deployment of DFMA and industrialised construction. A common aid to this approach is the digital configurator, a software platform that allows for users to explore options with predefined constraints such as kits-of-parts. This paper argues that this approach has fundamental limitations, in particular that (1) configurators are developed as standalone products to context-specific problems and that (2) current implementations lack a capability for downstream and upstream flows of information between design and construction processes of a building. This research proposes an open-source Common Configurator Framework that institutes a separation between the authoring of objects for the design processes and rules applied in the verification processes; a consequent standardisation and reuse of sets of parts (objects) and sets of rules (specifications) is demonstrated. Additionally, this work formalises the decoupling of the downstream part verification process and the upstream design requirements definition, so that a Common Specification-driven design and manufacturing process becomes possible. The work is supported by a reference implementation based on the proposed Framework with a series of case studies, prototypical

configurators and examples of verification/validation of pre-existing kits of parts against pre-authored specifications.

1. Introduction

The digital construction configurator is a common aid for the industrialised construction project which has gained traction in recent years. It allows a user to configure a construction project within a predetermined set of constraints, such as a kit of parts, an associated set of rules, and contextual factors. The wider AEC industry has recently seen a growing number of configurators being developed and publicised by practices, technology start-ups, software developers and others.

Recent studies have proposed a split of AEC configurators into the three categories of Planning, Design and Production, and observed that the large majority of the developed configurators falls into the first and second categories (Cao et al., 2021). We argue that this concentration is largely due to the nature of the work at early stages, which simplifies the structuring of a configurator, but also limits its scope and capabilities. In fact, attempts towards the development of an all-encompassing tool, which covers all the processes that take place in the AEC industry are often considered extremely challenging, due to the range and diversification of activities in design construction (Jensen et al., 2014).

Additionally, we observed that another critical point of existing AEC configurators is that they are developed in isolation of each other, often failing to recognize each other, therefore creating ad-hoc solutions to meet a specific, one-off need, with a limited leverage of existing functionality.

Our proposal was developed under the Construction Innovation Hub, a UK-Innovate government-funded initiative and targets these two issues by developing:

- A Common Configurator Framework (CCF), providing a common data schema on which multiple configurators can be built, with the capability of passing data from one configurator to the next. The CCF enables the implementation of configurators at all stages of the construction process – planning, design and production – moving away from the concept of an all-encompassing tool to an interconnected, modular suite of implementations that can work together to reach wider scope.
- A reference implementation of a chain of configurators, built following the above CCF, demonstrating the concept of interconnected configurators working across the supply chain.

2. Review of Configurators and platform-based design approaches in the AEC industry

The term “Configurator” is used broadly to encompass a variety of tools, interfaces and workflows, used to simplify the creation of complex systems, which is done by allowing users to select components from a set of alternatives - generally referred to as a kit-of-parts - in order to meet the end-users’ needs and requirements (Cao et al., 2021). Examples range from bespoke standalone desktop applications, web hosted apps or platforms, as well as extensions to existing proprietary technology and software, like plugins to other tools commonly used for design, delivery, and manufacturing. The configurator fulfils customer needs through the display and selection of products and delivery or production methods, enabling different levels of freedom.

Some authors (Haug et al., 2012; Cao et al., 2021) have proposed different implementation paths for Configurators under various Customer Order Decoupling Points (CODP). The CODP is a term describing the process or point in the supply chain where the activities are no longer driven by individual orders. The behaviour of processes upstream and downstream of the Customer Order

Decoupling Point is different: upstream processes are driven by forecast based planning information; downstream processes are instead driven by actual customer orders. The Open Reference Supply Chain Operations Domain formalizes these differences into several CODP strategies that are reflected in possible implementation paths for Configurators. This work considered in particular the following two common strategies:

- Engineer to Order (ETO): this enables customer-driven requirements – typically unknown or not designed at the time of product design/engineering – to be recorded on the order, therefore having the design/engineering finalized as part of the execution of the order. This is reflected in a Configurator implementation path which is founded on the definition of rules for the design and manufacturing.
- Configure to Order (CTO): the configuration of the product is part of the ordering process, and the product is therefore made after the order is received, entered and validated. Typical CTO processes work with modular products and with a menu-driven configuration process, with limited choices; non-menu-driven configuration processes are typically classified as Engineer-to-Order processes.

Recently, the need for configuration tools in the AEC industry has also been recognised by significant client bodies. For instance, the UK's Transforming Infrastructure Performance policy paper sets a focus on the implementation of a platform approach for social infrastructure, by highlighting that the technical solution for such an approach should include a digital catalogue, a set of "rules" and configuration tools. Such client-based interest has been anticipated in the industry through the development of several AEC commercial configurators:

- Project Frog, 2016: Focuses on interior design.
- Testfit, 2017: data-driven approach to smart urban planning.
- Archistar, 2018: assess and design construction sites.
- SpaceMaker, 2019: early-stage planning and design of real estate sites.
- PRISM, 2019: design of precision manufactured housing for London, UK.
- Hypar, 2019: platform for designing and sharing building systems.

It has been observed that these and other configurators can be fit into three main categories depending on the project phase that they mainly target. The Project Planning phase (1), targeted by Archistar and SpaceMaker, is largely the most tackled, as the speculative nature of early-stage planning is leaves the most space for configurators to show their benefits. For example, customers can be driven through the process of creating their desired apartment by real estate developers well before construction is under way. Furthermore, algorithms can be employed at this stage for early-stage, indicative estimates of performance indicators, which can later inform the actual design process. The Project Design phase (2) is also a widely applied situation for configurators; in this case, AEC professionals collaborate on e.g., detailed building models configured with pre-defined modules, and generally the outputs are converted to actionable data models, such as BIM models, that can be used for further development or analysis; Project Frog is an example of this. Finally, the Project Production phase (3) is where configurators are still largely absent. The aim in this case is generally to achieve mass customisation before the manufacturing process; typical outputs can be 3D BIM models, along with related data for fabrication machines, permit drawings and material listings.

Whilst most of the Configurators observed fit in specific project phase categories, some commercial configurators have taken an integrated approach across the planning and design process. For example, PRISM and Hypar propose to encompass the various stages of design, although they still fix certain decisions upfront: PRISM focuses on London housing, while Testfit mainly works on the urban planning scale. Few studies explore the appropriate theoretical foundation for truly integrated configurators (Cao et al., 2021), and all commercial configurators work in isolation from each other, without leveraging or integrating functionality that could be extracted from different implementations.

This gives the opportunity for the definition of a Common Configurator Framework that can help uniform the approaches to similar problems, enable inter-configurator communication and effectively tackle problems that encompass all stages of the construction process.

3. A Framework approach to Configurator design

In order to define a common understanding and an agreed definition of the term configurator within the context of industrialised design, manufacturing and construction within the AEC industry, we propose the definition of a configurator as: a typical process of Input - Computation – Output used to derive planning, design and manufacturing options constrained within a predetermined solution space (domain or kit-of-parts) and possibly controlled though a User Interface and Graphical Views.

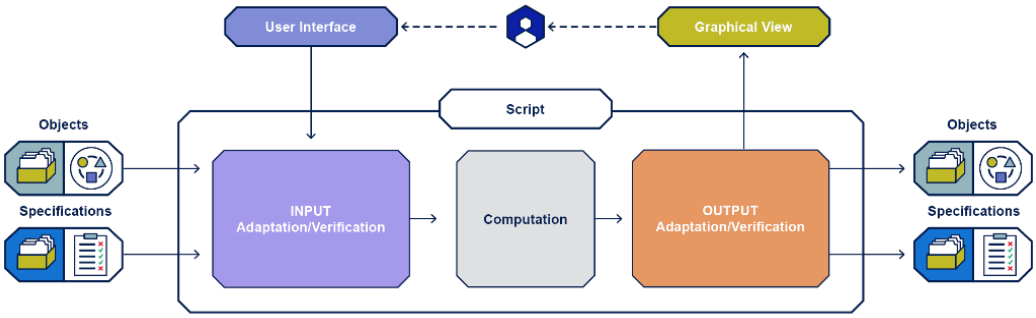

Figure 1. A Configurator: a typical process of Input - Computation – Output.

A configurator, as described above, is therefore a flexible mechanism for enabling wide engagement and interaction with a process from many individuals at different stage of the construction process. This can empower stakeholders across a project team, increasing collaboration and sharing know-how across traditional discipline/organizational boundaries. It is thus the potential for such configurators to be both easy to use and reusable, embed (and therefore make accessible) expertise and user requirements, and hence greatly increase the potential for automation. This poses some key challenges.

A first challenge is the transparent and robust communication of design assumptions and constraints between parties (and hence configurators), in addition to design data. A model that relies on sharing decentralized know-how must incorporate a mechanism for also sharing the appropriate and valid application of this know-how. We address this aspect by formalizing Objects and Object Specifications, which encode design data and design requirements in a Common Object Schema that allows any configurator that is part of the Framework to inter-communicate. This additionally enables distributed Verification and Validation to be achieved at transparently and robustly at scale.

Another key challenge is scalability. It is important to consider multiple stakeholder's user-requirements, whilst also designing for continuous reuse, customization and incorporation of future requirements or logic. Furthermore, a key characteristic of any configurator is that it embeds technical expertise. Reflecting on the highly distributed and wide-spread expert knowledge base that our industry relies upon, effective knowledge and know-how capture across all aspects of a project from conception to fabrication and operation is not a feasibly achievable task for a single team or limited number of individuals. It is therefore possible to conclude that a "single configurator" product cannot sufficiently address every detailed user requirement from all parties; instead, we suggest that a

scalable and distributed embedding and sharing of expertise can be obtained via a new approach that allows many configurators to connect together. In order to address this, this work refines the definition of Configurator, so that modularity and interoperability become possible.

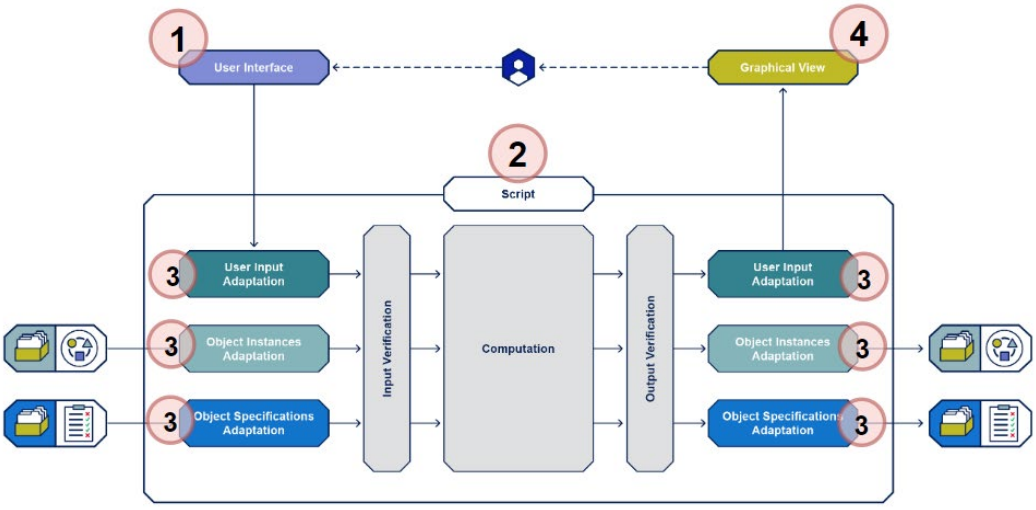

Figure 2. Configurator internal structure with its components labelled with numbers: (1) User Interface, (2) Script, (3) Adaptation, (4) Graphical View

The User interface is the module that is directly available to the User and that offers direct control over the inner logic of the configurator. This contrasts with the Script, which is imagined as a "black box" to the user; the script essentially contains the logic that performs the main computations of the configurator. Notice that the script performs some adaptation of the inputs and outputs: in order to allow complete freedom over the language of the script, we ask that the script becomes responsible for the adaptation of the format of the data – the object common schema – into the format that is consumable by the Configurator. Finally, the Graphical View provides another direct interaction with users, imagined as completely independent from the UI, allowing full freedom of choice between different software solutions for UI and output.

3.1. Objects in the Common Configurator Frameworks

We introduce the concept of objects as one of the core components in the framework necessary to facilitate exchange of information. An object is defined as any structured data (an agreed list of Properties, with Values) used to represent part of the product being configured. These objects (or collections of data) can then be shared in such a way that the configurator can operate upon them. Examples of objects could be anything from ducts, walls, beams, façade panels or component connections; products such as switches or light fittings; to abstract entities like levels or grids or indeed entire buildings.

The properties of objects can be anything from quantifiable metrics – such as length, height, mass, performance criteria like U-value, resistance to fire spread, global warming potential, etc. – through to less intrinsic quantities, such as cost or time. Object instances are taken by the Configurator as an input, to then potentially be configured or validated as part of its process of computation, and then finally the Configurator can also return new objects as part of its output.

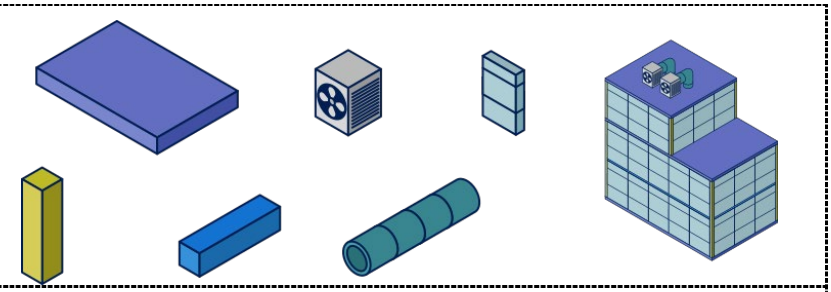

Figure 3. "Objects" can be defined representing building components as above, as well non-physical or abstract concepts such as grids, spaces or locations etc.

Following an Object-Oriented approach in formalising structured data within the framework, an Object is defined as a concrete instance of a certain Object Schema populated with specific Property Values such that it represents a specific element of the building or design. The framework itself places no constraints on the selection of schemas or ontological conventions other than that any selected schema must be consistent between the Objects and the Specifications utilised within any given chain of configurators. Thus, any classification system can be exploited, provided it can be used to define an unambiguous predefined set of entities (referred to here as objects) with named data fields (or properties).

Table 1. Example of properties for a "Column" object schema

Property name	Description	Possible values
Centreline	Geometrical centre line denoting the column position and axis	e.g., any Line
Cross section	Cross section of the column along the entire length	e.g., 300x500 mm
Material	Material the column is composed of	e.g., Concrete
Material Grade	Grade of the material the column is composed of	e.g., C25/30

For example, the "Column" schema can be given in JSON format, which is language-agnostic. In fact, the schema may in theory be defined using any language such as C# or JavaScript for instance:

```
1  {
2  "Centreline": {…}, // some geometrical object describing the centreline
3  "CrossSection": "300x500",
4  "Material": "Concrete",
5  "MaterialGrade": "C25/30"
6  }
```

Figure 4. An example of a Column object in JSON format. The properties are populated with values.

By leveraging this strategy, the Framework allows interchange to/from a variety of different Configurators, regardless of their internal implementation. This is a key concept, reinforced by the requirement of a Common Object Schema that allows coordination between different stakeholders across different trades.

```
1  public class Column
2  {
3      public IGeometry Centreline {get; set;}
4      public CrossSection CrossSection {get; set;}
5      public Material Material {get; set;}
6      public MaterialGrade MaterialGrade {get; set;}
7  }
```

Figure 5. An example of a Column schema in C#. This Column schema gives instances of Column objects

3.2. Object Specifications in the Common Configurator Framework

One of the core components in the framework necessary to facilitate exchange of information is the concept of Objects Specifications. An object specification defines a set of allowable values and constraints for the object's properties. This enables verification of object instances to be performed against the encoded predefined checks – determining if the instance of the object meets the specification and is valid or not.

The main objectives here are to formalize the concept of an Object Specification in a way that enables clear and easy authoring, maximises human readability during both the authoring, maximises human readability during the reviewing of both the object specifications and the results of the verification, maximises transparency and traceability of checks being performed to allow easy identification of any issues such as rules not respected, and what went wrong as well as conformation of compliance where that is the case.

A potential approach to a programmatic definition of a Specification compatible with the proposed Framework can be described as conditions that objects must respect. Thus, there is a way for the Specification to distinguish between the objects that must comply with the given Conditions (rules) and other objects that do not need to comply with those Conditions.

In other words, there must be a way for the Specification to filter the objects that must be verified. Once we have identified the objects that must be verified, the Filtered Objects, a condition can be applied to them, and they can be checked against this condition. We can say that a condition is simply a rule, which does not specify who (or what) is to respect this rule.

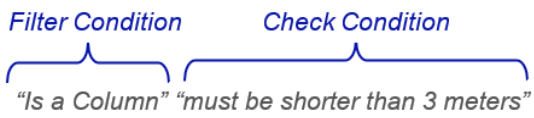

Figure 6. By decoupling the rule from the targets of the rule, we can see that this specification can be composed by two conditions: a filter condition that returns the "who" (Columns), followed by a check condition that gives the actual verification rule (shorter than)

There are two main features derived from the decoupling of the target of the rule from the rule itself. First, we maximise the modularity and reusability in the framework, including the ability to very transparently identify and reference individual rules in any specification or indeed have many rules applied to the same singular target. Second, we can then define who is to respect some rule by using another rule to define this target. This second point is critical to enable the specificity and control of targeted specification authoring and application necessary for use at scale on complex building projects. In conclusion, we can say that an Object Specification is defined as:

$$Object\ Specifications = \text{Filter Conditions} + \text{Check Conditions}$$

This effectively can be seen as a sort of "two-step" data validation process: first a filter (or set of filters) that returns the objects that should be checked, with second the checking rule(s) to be applied.

3.3. Defining a common Specification

The task of formalisation of a common mechanism for performing verification is important to enable adoption of industrialised processes at scale. In addition, defining a means to communicate the verification requirements or assumptions is needed for transparency, integrity and thus engagement across the many distributed parties essential in any supply chain.

In our framework we define a common Object Specification as a compact mechanism to encode rules that given objects can be checked against. These rules must be encoded in such a way that they can be read and understood by both human authors, checkers, and reviewers as well as directly actionable by a compatible digital configurator. Thus, a process of object verification can be performed with confidence by any party or tool that has access to sets of objects and specifications.

By structuring logical conditions and rules in this way, Object Specifications can thus be authored targeting any level of Component, Sub-Assembly, Interfaces between Sub-Assemblies or indeed against the overall Building or Project itself. The latter of which is a mechanism for encoding verifiable project level requirements as identified by the integrators, design team or client as part of the process of defining the need.

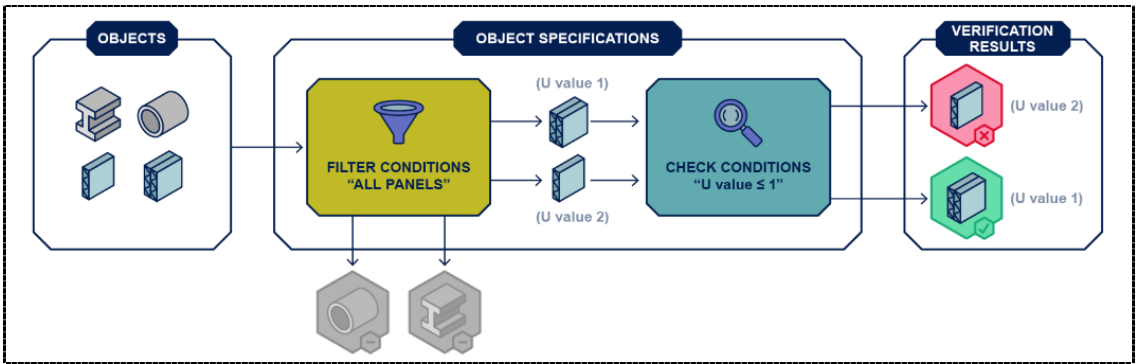

Figure 7. Illustration of an Object Specification as defined in our Configurator Framework. The combination of a Filter Condition followed by a Check Condition to enable transparent verification of objects. Here demonstrating verification of a façade panel based on its thermal insulating performance.

4. A Framework approach to Configurator design

The main theoretical contribution of the Framework towards the achievement of a Common Specification-driven design and manufacturing process is the formalisation of a decoupling between the downstream part verification process and the upstream design requirements definition. With a standard now established for distributed authoring and communication of both objects and object specifications, a critical new capability for the project team is now unlocked. That is the practice of recursive Verification and Validation (V&V) across an entire building or project. As illustrated in the figure below, a cascade of specification authorship targeting the various hierarchies of elements allows generation of a set of "coordinated specifications" as the output of a Design Requirements Process. These specifications in turn form the input to the Parts Verification stage of the project design and delivery process. This conceptual split between the design requirements definition and then verification against these specifications familiar form a Systems Engineering approach to design leads to two types of configurators: *Design Configurators* - generating a coordinated set of specifications and *Part Configurators* - operating on those provided specifications to ensure a fully verified and final design instance is defined.

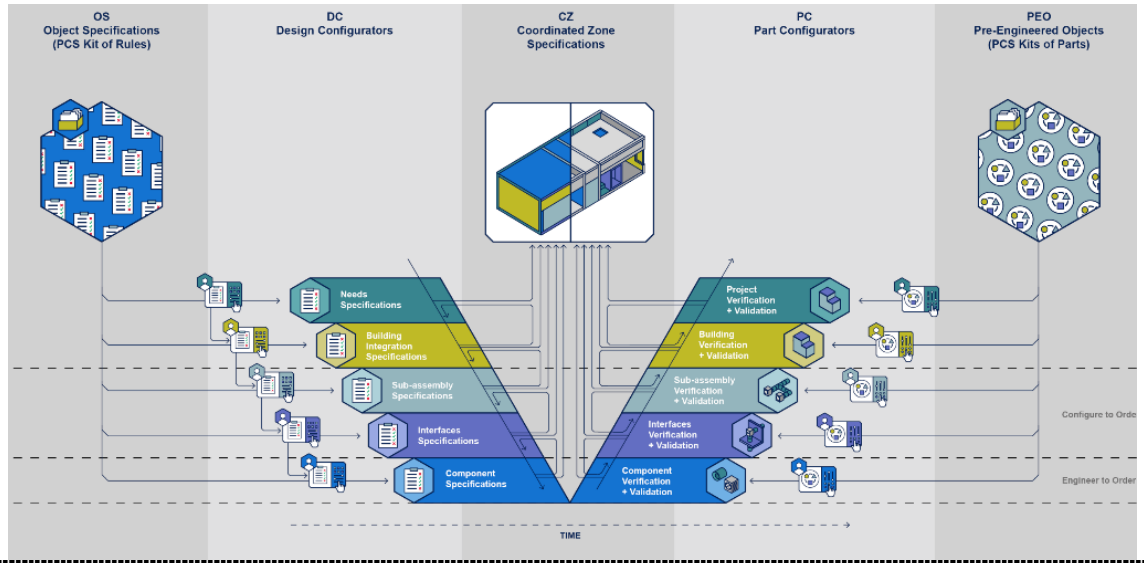

Figure 8. Design Configurators and Part Configurators interacting through common set of Coordinated Zone Specifications. The key feature to note here in this framework is the powerful decoupling of the downstream part verification process and the upstream definition of the design requirements. It is by communicating through these central Coordinated Zone Specifications that future extensibility and full flexibility across the supply chain is enabled.

This defined process of Part Configuration can therefore support a move to industrialised manufacturing, greater standardisation, and alternative production methodologies and ultimately adoption of a platform construction system approach to design procurement. Specifically, Part Configurators can utilise the Object Specification inputs to drive decision making and selection/optimisation of the parts themselves – rather than just simple part verification. Contrast engineer-to-order with configure-to-order in the figure below.

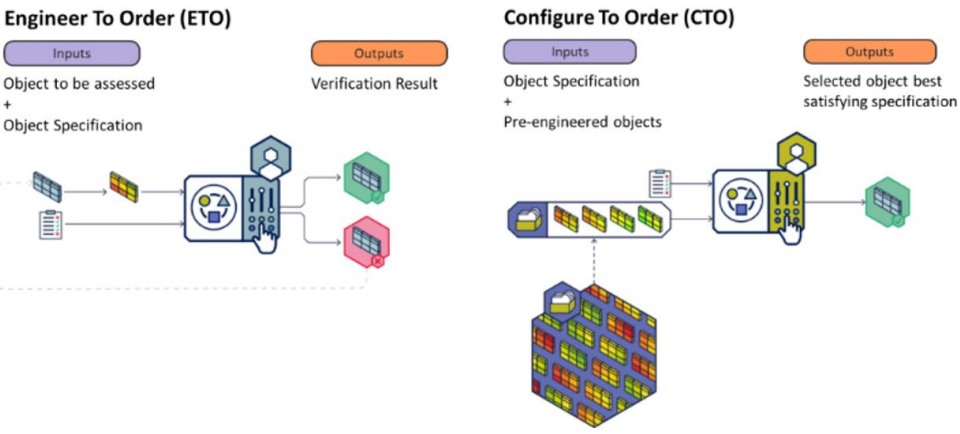

Figure 9. Engineer to Order, an object representing a proposed part is assessed against the required specification with the outcome being a verification result of either pass or fail. Configure to Order, the best performing part is selected from a library of pre-assessed parts with the output being a compliant object

5. A Reference Implementation of the Common Configurator Framework (CCF)

Whilst the need for connecting many discrete configurators has been recognised above, a reference implementation of the CCF has been the preliminary vehicle to test the main concept with three aims in mind. First, to demonstrate the configurator interconnectivity approach against practical applications proposed by both the Hub and the wider design streams. Second, the work must highlight the robustness of the Framework specification and the flexibility of the Digital Configurator approach. Finally, the reference implementation is to demonstrate how the Hub's platform-based design approach to both design requirements definition and subassembly and Part procurement processes can be facilitated.

In order to meet the aims, a set of discrete and distinct configurators, object specifications and encoded design decisions were implemented, as a strategic representative sample of the Hub requirements. The implementation of the configurators was based on the following observations. A complex design (i.e., a building) can be seen as a heterogeneous set of objects to which rules can be applied. This conceptualization allows to define rules for the global design while targeting individual objects (e.g., performance requirements for a modular façade system, while meeting the overall building embodied carbon requirements, etc.).

On these bases, the implementation of the Common Configurator Framework facilitated the critical chaining of individual configurators within different workflows, communicating through a centralized repository of coordinated Object Specifications. The many different configurators were intertwined in a non-linear way, representative of a typical platform-based design approach. Thanks to this interconnection, and by coordinating the specifications spatially, the framework implementation achieved the decoupling of the upstream design process from the Subassembly and Part procurement process. This allowed to exchange potential design components that could be verified and validated against centralized

performance requirements. The same centralized specifications were equally used to target objects across a variety of customisable workflows, allowing for flexibility in the process.

6. Conclusions, Limitations, and Future Research

This work proposed a Common Configurator Framework (CFF) that facilitates design and manufacturing best practice. The framework leverages a generalised approach via implemented Objects, Object Specifications and interconnected Configurators; it enables and encourages a modular and incremental implementation of a digital platform construction system, extendable with further modules over time. Therefore, the work allows any configurator developed in accordance with or adjusted to adopt the CCF to be truly integrated with any other CCF configurator. This in turns enables true integrated approaches to both Design and Construction workflows. The work demonstrates how this approach can enable an incremental transformation of the construction industry, by progressively moving towards a platform construction system model, utilising simultaneously Configure to Order, Engineer to Order and traditional methods across both sub-assemblies and different components of the same project.

At the same time, a number of factors that limit the adoption potential of the CCF have been identified. In terms of governance, further work and collaborations are needed to facilitate the encoding of specifications and objects. This would benefit from the standardisation and streamlining of the verification process of parts, through the introduction and adoption of Product Data Templates (PDTs) and the adoption of dedicated PDT and product instancing authoring tools, which derive from an industry-wide consensus framework. A potential limitation towards that direction would be the current inconsistency of the digitalisation process and the level of digital literacy across the AEC industry. In parallel, the adoption of the proposed encoding of Rules should be driven by further work on the Rulebook. Rules can be extracted and encoded as Specifications, as described, by distinguishing Filter Conditions and Check Conditions. This encoding will enable the targeting of any heterogeneous set of objects and the flexible decoupling of configurators and workflows.

Furthermore, from a technical standpoint, the CCF would benefit from case studies that explore the relationships between human-driven and automated Validation and Verification processes. This would allow to address a wider spectrum of scenarios, and of increasing complexity, going beyond the simple numeric verification of certain parameters; for example, how the acoustic performance of an interface between a curtain wall and its supporting structural frame is affected by different design choices.

References

[1] Cao J and Hal Dl 2019 An Overview of Configurations for Industrialized Construction: Typologies, Customer Requirements, and Technical Approaches, (2019 European Conference on Computing in Construction, July 2019), 295–303.

[2] Cao J et al., 2021 ‘Cross-Phase Product Configurator for Modular Buildings Using Kit-of-Parts’, Automation in Construction **123** (March): 103437.

[3] Haug A, Hvam L and Mortensen N H 2012 Definition and Evaluation of Product Configurator Development Strategies, Computers in Industry **63**, no. 5 (June): 471–81.

[4] Smiding E, Gerth R and Jensen P 2016 Developing Product Configurators in the AEC

Industry', in ICCREM 2016 (International Conference on Construction and Real Estate Management, Edmonton, Canada: American Society of Civil Engineers), 135–45.

[5] Jensen P et al. 2014 Developing Products in Product Platforms in the AEC Industry, in Computing in Civil and Building Engineering (2014 International Conference on Computing in Civil and Building Engineering, Orlando, Florida, United States: American Society of Civil Engineers, 2014), 1062–69.

[6] Jensen P 2014 Configuration of Platform Architectures in Construction (Luleå University of Technology).

[7] Haug A, Hvam L and Mortensen N H 2012 Definition and Evaluation of Product Configurator Development Strategies, Computers in Industry 63, no. 5 (June): 471–81.

[8] Open Reference, Customer Order Decoupling Point (CODP), https://orwiki.org/t:CODP, last visited 14/01/2022.

[9] UK Infrastructure and Projects Authority, Transforming Infrastructure Performance: Roadmap to 2030, 13 September 2021, https://www.gov.uk/government/publications/transforming-infrastructure-performance-roadmap-to-2030/transforming-infrastructure-performance-roadmap-to-2030, last accessed 14/01/2022.